**R Functions**

A function is a set of statements organized together to perform a specific task. R has many in-built functions, and the user can create their own functions. In R, a function is an object so that R interpreter can pass control to the function, along with arguments that may be necessary for the function to accomplish the actions. The function in turn performs its task and returns control to the interpreter as well as any result which may be stored in other objects.

Functions allow you to automate common tasks in a more powerful and general way than copying and pasting. Writing a function has three big advantages over using copy-and-paste:

* You can give a function an evocative name that makes your code easier to understand.
  + As requirements change, you only need to update code in one place, instead of many.
  + You eliminate the chance of making incidental mistakes when you copy and paste.
  + (i.e., updating a variable name in one place, but not in another).

An R function is created by using the keyword function. The basic syntax of an R function definition is as follows:

function\_name <- function(arg\_1, arg\_2, ...)

{

Function body

}

**Function Components**

* Function Name: This is the actual name of the function. It is stored in R environment as an object with this name.
* Arguments: An argument is a placeholder. When a function is invoked, you pass a value to the argument. Arguments are optional; that is, a function may contain no arguments. Also, arguments can have default values.
* Function Body: The function body contains a collection of statements that defines what the function does.
* Return Value: The return value of a function is the last expression in the function body to be evaluated.

**Examples of inbuilt functions**

# Create a sequence of numbers

print(seq(12,21))

# Find mean of numbers from 25 to 82.

print(mean(25:82))

# Find sum of numbers from 1 to 20.

print(sum(1:20))

**User-defined Function**

We can create user-defined functions in R. They are specific to what a user wants and once created they can be used like the built-in functions. Below are examples of how a function is created and used in different ways.

# Create a function to print squares of numbers in sequence.

Square <- function(a)

{

for (i in 1:a)

{

b <- i^2

print(b)

}

}

**Calling a Function without an Argument**

# Create a function without an argument.

Square <- function()

{

for(i in 1:10)

{

print(i^2)

}

}

Square()

**# Create a function with arguments.**

maths <- function(a,b,c)

{

x <- a\*b+c^2

print(x)

}

Maths(1,2,3)

Maths(a=1,b=2,c=3)

**# Create a function with arguments**.

default <- function(a = 3,b =6)

{ z <- a\*b

print(z)

}

Default()

**Lazy Evaluation of Function**

Arguments to functions are evaluated lazily, which means so they are evaluated only when needed by the function body.

# Evaluate the function without supplying one of the arguments

# Create a function with arguments.

new <- function(a, b)

{ print(a^2)

print(a)

print(b).

new(2)

**R strings**

Any value written within a pair of single quote or double quotes in R is treated as a string. Internally R stores every string within double quotes, even when you create them with single quote.

**Rules Applied in String Construction**

• The quotes at the beginning and end of a string should be both double quotes or both single quote. They cannot be mixed.

• Double quotes can be inserted into a string starting and ending with single quote.

• Single quote can be inserted into a string starting and ending with double quotes.

• Double quotes cannot be inserted into a string starting and ending with double quotes.

• Single quote cannot be inserted into a string starting and ending with single quote.

**Examples of valid strings**

a <- 'Start and end with single quote’.

print(a)

b <- "Start and end with double quotes”.

print(b)

c <- "single quote ' in between double quotes"

print(c)

d <- 'Double quotes " in between single quote'

print(d)

**Examples of Invalid Strings**

e <- 'Mixed quotes"

print(e)

f <- 'Single quote ' inside single quote'

print(f)

g <- "Double quotes " inside double quotes"

print(g)

Many strings in R are combined using the paste() function. It can take any number of arguments to be combined.

The basic syntax for paste function is :

paste(..., sep = " ", collapse = NULL)

• ... represents any number of arguments to be combined.

• sep represents any separator between the arguments. It is optional.

• collapse is used to eliminate the space in between two strings. But not the space within two words of one string.

a <- "Hello"

b <- 'How'

c <- "are you? "

print(paste(a,b,c))

print(paste(a,b,c, sep = "-"))

print(paste(a,b,c, sep = "", collapse = ""))

Numbers and strings can be formatted to a specific style using format()function.

Counting number of characters in a string -ncahr() function.

Changing the case -toupper() & tolower() functions.

Extracting parts of a string -substring() function